测试报告
version 1.0.0

修订历史

日期	版本	作者	描述
8 月 21 日	1.0.0	GA237	完成 Jumper 类大部分方法的测试

目录

1 测试计划

（1）act 方法

act 方法是 Jumper 功能实现的核心方法。因此着重对 act 函数进行测试，测试包含以下的情况：

a. 当 Jumper 前面的两个格子都是空的时候 Jumper 的行为；

b. 当 Jumper 前面的第一个格子有石头(Rock)，第二个格子为空的时候 Jumper 的行为；

c. 当 Jumper 前面的第一个格子为空，第二个格子有石头的时候 Jumper 的行为；

d. 当 Jumper 前面的两个格子都有石头的时候 Jumper 的行为；

e. 当 Jumper 前面的的两个格子范围内有边界的时候 Jumper 的行为；

f. 当 Jumper 前面有花朵(Flower)的时候 Jumper 的行为。

（2）jump 方法

jump 方法会强制性地往当前方向移动两格，且并不判断是否越过边界，或两格前面有石头。（因为相关判断由 canMove 完成）

故简单的对函数调用前后的坐标进行对比即可。另外，当使用 jump 方法，使得 jumper 跳出边界的时候，jumper 应当从该 grid 删除。

（3）move 方法

与 jump 方法类似，不同点在于 move 方法只让 Jumper 向当前方向移动两格，故与 jump 方法的测试类似。

（4）canJump 方法

判断 Jumper 前面的第 2 格是否可以到达（格子中有障碍物的时候不行，除了是花朵）。测试包括：

a. 前面的第 2 格有石头

b. 前面的第 2 格没有石头

（5）canJump 方法

判断 Jumper 前面的第 2 格是否可以到达（格子中有障碍物的时候不行，除了是花朵）。测试包括：

a. 前面的第 2 格有石头

b. 前面的第 2 格没有石头

（6）canMove 方法

判断 Jumper 前面的第 1 格是否可以到达（格子中有障碍物的时候不行，除了是花朵）。测试包括：

a. 前面的第 1 格有石头

b. 前面的第 1 格没有石头

（7）Jumper(Color jumperColor)方法

测试是否能设置正确的颜色

（8）canPutOntoLocation 方法

对于放有不同 Actor 的坐标，测试其返回值是否正确。

2 用例 1

测试目的：

1. 测试当 Jumper 前面的两个格子都是空的时候 Jumper 调用 act 方法的行为是否正确
2. 测试当 Jumper 前面的两个格子的范围内有边界的时候 Jumper 调用 act 方法的行为是否正确

2.1 测试用例

测试方案：

1. 创建一个 Jumper 放进网格内，坐标为(3, 5)
2. 连续执行两次 act 方法，每次执行后判断 Jumper 当前的位置

预期结果：

第一次调用 act 方法，其坐标应为(1, 5)；

第二次调用 act 方法，其坐标应为(0, 5)

测试代码如下：

```java
/**
 * Test the condition which both forward two cells can be reached
 */
@Test
public void testAct1() {
    Jumper alice = new Jumper();
    world.add(new Location(3, 5), alice);
    alice.act();
    assertEquals(new Location(1, 5), alice.getLocation());
    alice.act();
    assertEquals(new Location(0, 5), alice.getLocation());
}
```

2.2 测试结果

（测试结果截图在最后给出）

2.3 结果分析

结果与预期一致。这是因为第一次的行动的时候毫无障碍物，因此可以直接向上移动两格，此时离边界还有一格，故第二次行动的时候最多只能向上移动一格。

3 用例 2

测试目的：

测试当 Jumper 前面的第 2 个格子有障碍物的时候 Jumper 调用 act 方法的行为是否正确。

3.1 测试用例

测试方案：

1. 创建一个 Jumper 放置于网格内，坐标为(5, 5)
2. 创建一个 Rock 放置与网格内，坐标为(3, 5)
3. 调用一个 act 方法，判断 Jumper 的当前坐标

预期结果：

act 方法调用结束后，Jumper 的坐标应为(4, 5)

测试代码如下：

```
/**
 * Test the condition if the location in front of it is empty,
 * but the location two cells in front contains a rock
 */
@Test
public void testAct2() {
    Jumper alice = new Jumper();
    world.add(new Location(5, 5), alice);
    world.add(new Location(3, 5), new Rock());
    alice.act();
    assertEquals(new Location(4, 5), alice.getLocation());
}
```

3.2 测试结果

（测试结果截图在最后给出）

3.3 结果分析

结果与预期一致，当前方有障碍物使得 Jumper 不能到达前方第二格的时候，若前方第一格可达，那么就只前进一格。

4 用例 3

测试目的：

测试当 Jumper 前面的第 1 个格子有障碍物的时候 Jumper 调用 act 方法的行为是否正确。

4.1 测试用例

测试方案：

1. 创建一个 Jumper 放置于网格内，坐标为(5, 5)

2. 创建一个 Rock 放置与网格内，坐标为(4, 5)

3. 调用一个 act 方法，判断 Jumper 的当前坐标

预期结果：

act 方法调用结束后，Jumper 的坐标应为(3, 5)

测试代码如下：

```
/**
 * Test the condition if the location two cells in front of it is empty,
 * but the location in front contains a flower or a rock
 */
@Test
public void testAct3() {
    Jumper alice = new Jumper();
    world.add(new Location(5, 5), alice);
    world.add(new Location(4, 5), new Rock());
    alice.act();
    assertEquals(new Location(3, 5), alice.getLocation());
}
```

4.2 测试结果

（测试结果截图在最后给出）

4.3 结果分析

结果与预期一致，当前方第 2 格没有障碍物的时候，无论第 1 格有没有障碍物，Jumper 都能正常到达第 2 格。

5 用例 4

测试目的：

测试当 Jumper 前面的 2 个格子均有障碍物的时候 Jumper 调用 act 方法的行为是否正确。

5.1 测试用例

测试方案：

1. 创建一个 Jumper 放置于网格内，坐标为(5, 5)
2. 创建一个 Rock 放置与网格内，坐标为(3, 5)
3. 创建一个 Rock 放置与网格内，坐标为(4, 5)
4. 调用一个 act 方法，判断 Jumper 的当前坐标

预期结果：

act 方法调用结束后，Jumper 的坐标应为(5, 5)

测试代码如下：

```
/**
 * Test the condition if both two location contain a rock
 */
@Test
public void testAct4() {
    Jumper alice = new Jumper();
    world.add(new Location(5, 5), alice);
    world.add(new Location(4, 5), new Rock());
    world.add(new Location(3, 5), new Rock());
    alice.act();
    assertEquals(new Location(5, 5), alice.getLocation());
}
```

5.2 测试结果

（测试结果截图在最后给出）

5.3 结果分析

结果与预期一致，当前方的两格都有障碍物的时候，Jumper 就不能移动，停留在原地。

6 用例 5

测试目的：

测试当 Jumper 前面的第 2 个格子有花朵的时候 Jumper 调用 act 方法的行为是否正确。

6.1 测试用例

测试方案：

1. 创建一个 Jumper 放置于网格内，坐标为(5, 5)
2. 创建一个 Rock 放置与网格内，坐标为(4, 5)
3. 创建一个 Flower 放置与网格内，坐标为(3, 5)
4. 调用一个 act 方法，判断 Jumper 的当前坐标

预期结果：

act 方法调用结束后，Jumper 的坐标应为(3, 5)

测试代码如下：

```
@Test
public void testFlower() {  //Test Jumper weather can take over the flower
    Jumper alice = new Jumper();
    world.add(new Location(5, 5), alice);
    world.add(new Location(3, 5), new Flower());
    world.add(new Location(4, 5), new Rock());
    alice.act();
    assertEquals(new Location(3, 5), alice.getLocation());
}
```

6.2 测试结果

（测试结果截图在最后给出）

6.3 结果分析

结果与预期一致，花朵不算障碍物，故依然可以到达。

7 用例 6

测试目的：

测试 canMove 和 canJump 方法的功能是否正确实现。

7.1 测试用例

测试方案：

1. 创建一个 Jumper 放置于网格内，坐标为(5, 5)
2. 创建一个 Flower 放置与网格内，坐标为(3, 5)
3. 创建一个 Rock 放置与网格内，坐标为(4, 5)
4. 调用 canMove 方法，将返回值与预期值进行比较
5. 调用 canJump 方法，将返回值与预期值进行比较

预期结果：

1. canMove 方法的返回值为 false
2. canJump 方法的返回值为 true

测试代码如下：

```
@Test
public void testCanMJ() {  //Test canMove(), canJump()
    Jumper alice = new Jumper();
    world.add(new Location(5, 5), alice);
    world.add(new Location(3, 5), new Flower());
    world.add(new Location(4, 5), new Rock());
    assertFalse(alice.canMove());
    assertTrue(alice.canJump());
}
```

7.2 测试结果

（测试结果截图在最后给出）

7.3 结果分析

结果与预期一致，因为 Jumper 前方第 1 格有石头，故 Jumper 不能向前移动一格，而第 2 格只是花朵，可以移动到第二格，故 canJump 是 true 的，canMove 是 false 的。

8 用例 7

测试目的：

测试 testCanPutOntoLocation 方法是否正确实现。

8.1 测试用例

测试方案：

1. 创建一个 Jumper 放置于网格内，坐标为(5, 5)
2. 创建一个 Flower 放置与网格内，坐标为(4, 5)
3. 调用 testCanPutOntoLocation 方法，参数为 Location(4, 5)，得到返回值
4. 创建一个 Rock 放置与网格内，坐标为(3, 5)
5. 调用 testCanPutOntoLocation 方法，参数为 Location(3, 5)，得到返回值
6. 创建一个 Jumper 放置于网格内，坐标为(6, 6)
7. 调用 testCanPutOntoLocation 方法，参数为 Location(6, 6)，得到返回值

预期结果：

1. 返回值为 true
2. 返回值为 false
3. 返回值为 false

测试代码如下：

```
@Test
public void testCanPutOntoLocation() {
    Jumper alice = new Jumper();
    world.add(new Location(5, 5), alice);

    world.add(new Location(4, 5), new Flower());
    assertTrue(alice.canPutOntoLocation(new Location(4, 5)));

    world.add(new Location(3, 5), new Rock());
    assertFalse(alice.canPutOntoLocation(new Location(3, 5)));

    Jumper bob = new Jumper();
    world.add(new Location(6, 6), bob);
    assertFalse(bob.canPutOntoLocation(alice.getLocation()));
}
```

8.2 测试结果

（测试结果截图在最后给出）

8.3 结果分析

结果与预期一致，因为 Jumper 只能移动到空白的格子或者是有花朵的格子。

9 用例 8

测试目的：

测试带参构造函数的正确性。

9.1 测试用例

测试方案：

1. 创建一个 Jumper，其构造函数参数为 Color.ORANGE
2. 调用 getColor 方法获得 Jumper 的颜色，与开始时设置的颜色进行比较。

预期结果：

1. getColor 方法的返回值为 Color.ORANGE

测试代码如下：

```
@Test
public void testJumperColor() {
    Jumper alice = new Jumper(Color.ORANGE);
    assertEquals(Color.ORANGE, alice.getColor());
}
```

9.2 测试结果

（测试结果截图在最后给出）

9.3 结果分析

结果与预期一致，颜色属性被正确地修改。

10 用例 9

测试目的：

测试 jump 方法的正确性。

10.1 测试用例

测试方案：

1. 创建一个 Jumper 放置于网格内，坐标为(5, 5)
2. 调用 jump 方法，将 jumper 当前坐标与预期值进行比较
3. 调用 jump 方法，将 jumper 当前坐标与预期值进行比较
4. 调用 jump 方法，判断 jumper 的 grid 属性是否为 Null

预期结果：

1. 第一次调用方法，坐标为(3, 5)
2. 第二次调用方法，坐标为(1, 5)
3. 第三次调用方法，jumper 的 grid 属性为 null

测试代码如下：

```
@Test
public void testJump() {
    Jumper alice = new Jumper();
    world.add(new Location(5, 5), alice);
    alice.jump();
    assertEquals(new Location(3, 5), alice.getLocation());
    alice.jump();
    assertEquals(new Location(1, 5), alice.getLocation());

    // out of grid
    alice.jump();
    assertEquals(null, alice.getGrid());
}
```

10.2 测试结果

（测试结果截图在最后给出）

10.3 结果分析

结果与预期一致，当 jumper 到了一个非法位置后，会将自己从网格中删除，故其 grid 和 location 属性都变为 null 了。

11 用例 10

测试目的：

测试 turn 方法的正确性。

11.1 测试用例

测试方案：

1. 创建一个 Jumper 放置于网格内，坐标为(5, 5)
2. 调用 turn 方法，将 jumper 当前朝向的方向角度对 45 取模

预期结果：

取模后结果为 0

测试代码如下：

```
@Test
public void testTurn() {
    Jumper alice = new Jumper();
    world.add(new Location(5, 5), alice);
    alice.turn(); // trun to random degree
    assertTrue(alice.getDirection() % 45 == 0);
}
```

11.2 测试结果

（测试结果截图在最后给出）

11.3 结果分析

结果与预期一致，由于 turn 方法是随机转向的，故不能准确地判断角度，对其取模可以有一个大致的判断。

最终结果截图

Unit Test Results.

Designed for use with JUnit and Ant.

Class JumperTest

Name	Tests	Errors	Failures	Skipped	Time(s)	Time Stamp	Host
JumperTest	10	0	0	0	0.141	2015-08-21T08:39:10	vinzor-virtual-machine

Tests

Name	Status	Type	Time(s)
testCanMJ	Success		0.025
testAct1	Success		0.000
testAct2	Success		0.000
testAct3	Success		0.008
testAct4	Success		0.001
testJump	Success		0.000
testTurn	Success		0.001
testCanPutOntoLocation	Success		0.001
testFlower	Success		0.001
testJumperColor	Success		0.001